Development of an IoT System with Smart Charging Current Control for Electric Vehicles

Ruben A. Sousa
CMEMS-Uminho Center
University of Minho
Guimaraes, Portugal
a70013@alunos.uminho.pt

Vitor Monteiro
Centro ALGORITMI
University of Minho
Guimaraes, Portugal
vmonteiro@dei.uminho.pt

Joao C. Ferreira
ISTAR-IUL Center
ISCTE-IUL
Lisbon, Portugal
joao.carlos.ferreira@iscte-iul.pt

Andres A. Nogueiras Melendez
Departamento de Tecnología Electrónica – University of Vigo
Vigo, Spain
aaugusto@uvigo.es

Joao L. Afonso
Centro ALGORITMI
University of Minho
Guimaraes, Portugal
jla@dei.uminho.pt

Jose A. Afonso
CMEMS-Uminho Center
University of Minho
Guimaraes, Portugal
jose.afonso@dei.uminho.pt

Abstract—**This paper presents the development and test of an Internet of Things (IoT) system for monitoring and control of electric vehicles. The IoT architecture, which was developed using the Firebase platform, allows the synchronization of the vehicles' data to the online server, as well as the access to the data outside of the vehicle, though the Internet. The smart charging system proposed in this paper allows the control of the electric vehicle's battery charging current in real time, based on the demand at the residence (home current), which is measured using a residential wireless sensor network (WSN). An Android mobile app was developed to access the vehicle's data. This app communicates with the wireless sensor nodes of an intra-vehicular wireless sensor network (IVWSN), which was developed using the Bluetooth Low Energy (BLE) protocol. A real time notification system was also implemented to alert users about certain events, such as low battery and full battery charge. The main features of the proposed IoT system are validated through experimental results.**

Keywords— ***Internet of Things, Smart Charging System, Electric Vehicles, Wireless Sensor Networks, Mobile Sensing.***

I. Introduction

The steadily increase in the availability of mobile Internet access throughout the world and its diminishing cost led to widespread generation and use of mobile data. This new reality, along with recent advances in wireless sensor networks (WSNs) [1] and mobile sensing [2][3], opens up an enormous range of possibilities and projects related to the Internet of Things (IoT) [4].

With characteristics such as low cost, low energy consumption [5], low latency and high reliability, Bluetooth Low Energy (BLE) takes a leading position for the implementation of intra vehicular wireless sensor networks (IVWSN) [6], as well as to enable the development of the IoT [7]. In a previous work, BLE was shown to achieve good reliability inside of vehicles [8], despite its metallic parts. In [9], Siekkinen et al. concluded that BLE presents lower energy consumption compared to the alternative IEEE 802.15.4/ZigBee wireless network technology. Lin et al. [10] demonstrated that BLE outperforms ZigBee inside of vehicles under Wi-Fi interference, which can be explained by the use of frequency hopping spread spectrum (FHSS), which increases the robustness against interference and fading because the transmissions are spread through the available 2.4 GHz band, whereas ZigBee transmissions are restricted to a fixed channel. Another advantage of BLE over ZigBee is the native hardware and software support of the protocol provided by most of the modern mobile devices.

This paper describes the development of a smart IoT system that builds upon a previous work [8], with the provision of several new features. The previous system was composed by an IVWSN, which collected real time data from multiple sensor nodes inside of the vehicle, and a mobile app, which presented the collected data to the user. The IVWSN was implemented using the BLE protocol (version 4.0), due to its referred advantages, and the hardware/software of the sensor nodes was based on the CC2540 platform, from Texas Instruments.

The system described in this paper extends the capabilities of the mobile app, providing communication with a remote server, in an IoT-based architecture, allowing the data collected from the vehicle to become available to other devices through the Internet. In the developed prototype, the mobile app also communicates with a BLE-based residential wireless sensor network, allowing the provision of a smart charging solution for electric vehicles, where the battery charging current is controlled in real time based on the measured home current.

In synthesis, the proposed system presents four main enhancements compared with the previous work. The first one is the integration of an IoT server database, where the collected data may be stored and accessed. The second one is the enhancement of the mobile app, which now has two operation modes: as a gateway, inside of the vehicle, or as a mobile client, anywhere in the world with a connection to the Internet. In the gateway mode, the mobile app inherits the capabilities available in the previous work, namely the communication with the IVWSN sensor nodes and the role of vehicle's Human-Machine Interface (HMI), but the interface was redesigned and now the mobile app is also able to exchange data with the IoT server. The client mode, on the other hand, allows the monitoring and control of the vehicles' parameters real time through access to the server database. The third improvement is the development of a new IVWSN network prototype, which replaces the former one with a newer version of the BLE protocol (v4.2) [11] and a more recent hardware/software platform, from a different manufacturing company (Cypress Semiconductor). The fourth one is the development of a residential WSN, based on the same Cypress BLE platform and a Raspberry Pi gateway, which is also integrated in the architecture of the proposed smart IoT system.

II. System Implementation

The main components of the architecture of the developed IoT system, shown in Fig. 1, are:

- a Firebase server database;
- an Android mobile app, which may act as the vehicle's gateway/HMI or a mobile client;
- an IVWSN composed by the vehicle's gateway and BLE sensor nodes;
- a residential WSN, which integrates a BLE/Wi-Fi gateway (Raspberry Pi) and a sensor node to measure the home current.

The implementation of each of these components is described in the following sections.

Fig. 1 shows only the network sensor nodes that were implemented in the presented prototype, but the system is scalable, allowing both the IVWSN and the residential WSN, which are both based on the BLE protocol, to accommodate more sensor nodes. The scalability of BLE was studied in a previous work [12], which demonstrates that the BLE protocol is able to support a high number of sensor nodes generating high data rate traffic.

The system prototype was developed and tested using an electric vehicle developed by the Group of Energy and Power Electronics (GEPE) of University of Minho, Portugal [13], but it was designed to support several vehicles, and may accommodate other types of sensors.

III. Database Development

This section describes the Firebase configuration for use in the developed system, the database structure that was defined for this system and the associated implemented functions. The Firebase platform was chosen for this work due to lower costs for subscription of the required services in a comparative analysis with Microsoft Azure and Amazon Web Services (AWS) [14].

A. Firebase Configuration

Before starting the configuration, it is necessary to create a project in Firebase that identifies the mobile app. The method chosen to authenticate the users of the developed Android app was through the Google account. Since all Android users are required to have a Google account to download apps, this method makes it easier to authenticate, with a simple button that opens the list of Google accounts on the device, instead of the alternative of entering a login/password.

For Google Play services to recognize the mobile application that is attempting to use the authentication service, it is necessary to add the SHA-1 digital signature of the

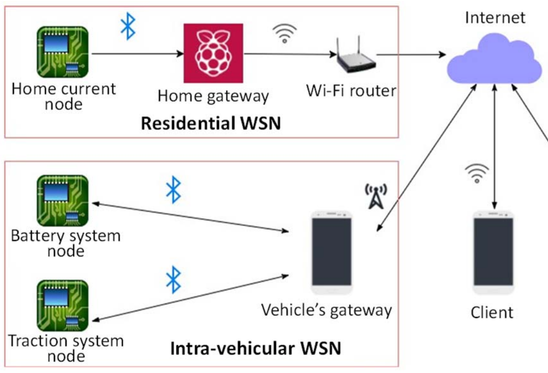


Fig. 1. Architecture of the developed IoT system for electric vehicles.

certificate that is used to sign the APK file. This digital signature can be obtained with the keytool utility, which is included in the Java installation, or through an option in Android Studio. The SHA-1 digital signature needs to be added to the application settings in Firebase. Then, in the app, it is necessary to request an access token to the Google service and use it to authenticate with Firebase.

B. Database Structure

The structure of the database of the developed system is represented by a JavaScript Object Notation (JSON) file that contain collections of objects. The first step in structuring this file was to identify the data that needs to be collected: user data, notification settings, battery system data, traction system data, vehicle location and electric current consumption at home.

The first key of the JSON file is "users", and contains a list with unique identifiers of each user, which are randomly generated by Firebase. Inside each user field, there are the "vehicles", "notifications" and "homeCharging" fields, which represent the added user vehicles, the notifications for the user and the charging system data, respectively. For the "vehicles" field, the following data is stored: license number, model, battery charging options (manual or smart), vehicle location, battery system data and traction system data.

For the "notifications" field, two Boolean values are stored, related to the status of the electric vehicle's battery charge: full charge and low charge, as well as unique tokens of the registered mobile devices to make it possible to send notifications. For the charging system field, only the home current, in Amperes, and the respective timestamp, in ms, are added. In order to allow only authorized users to access their vehicle data, security rules have been defined in Firebase. Only the authenticated users can access the data from their vehicles. The authentication is based on the "auth.uid" property, with separate fields defined in the rules for read and write data.

C. Implemented Functions

For the developed prototype, it was only necessary to implement one function in Firebase. This function is used to send notifications according to the battery charge level of the electric vehicle if this option is enabled by the user in the mobile app settings (inside the user profile configuration screen). The function is called whenever the battery charge value is updated, as shown in Fig. 2. When the charge level

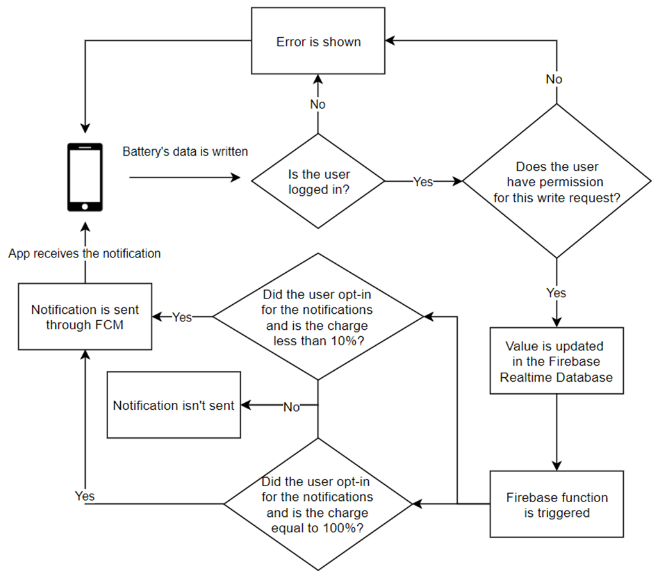

Error is shown
No
No
Battery's data is written
Is the user logged in?
Yes
Does the user have permission for this write request?
App receives the notification
Yes
Notification is sent through FCM
Yes
Did the user opt-in for the notifications and is the charge less than 10%?
Value is updated in the Firebase Realtime Database
Notification isn't sent
No
Yes
Did the user opt-in for the notifications and is the charge equal to 100%?
Firebase function is triggered

Fig. 2. Flow of events when the battery charge value is updated.

becomes less than 10% (this value can be changed in the app settings) or reaches 100%, a notification is sent to the user using the Firebase Cloud Messaging (FCM) platform.

IV. Development of the BLE Sensor Nodes

This section describes the development of the firmware of the implemented wireless sensor nodes. Two sensor nodes, for the battery system and the traction system, are part of the IVWSN, whereas a third node is used in the residential WSN. All the sensor nodes were configured as BLE peripheral stations. The role of BLE central station, on the other hand, was assigned to vehicle's gateway (an Android tablet) in the case of the IVWSN, and the home gateway (a Raspberry Pi), in the case of the residential WSN. The CY8CKIT-042-BLE-A BLE development kit was used for the development of the peripheral stations. Each kit includes a development board (BLE Pioneer), one BLE module and a BLE USB dongle. The BLE modules were used for the sensor nodes. The development of the C code for the microcontrollers of the BLE modules was made using the PSoC Creator 4.1 IDE.

A. Components Configuration

In the context of the developed prototype, all the peripheral stations in the vehicle integrate the following components: BLE, UART (Universal Asynchronous Receiver/Transmitter), Timer and Bootloader. The BLE component is used to configure the Bluetooth protocol parameters, such as the contents of the advertisement packets, the connection interval, the Generic Attribute Profile (GATT) characteristics and the BLE notifications. The UART component is used by the respective peripheral station to collect data from the battery system or the traction system. The Timer component is used to generate an interrupt after a certain period, in order to send the collected data to the central station. The default value was configured to 500 ms in the prototype, but it can be changed by a command received from the mobile app. Finally, the Bootloader manages the writing of code or data to flash memory of the device and allows the use the RX and TX lines of the UART component via USB. These components can be added to the schematic file of the firmware project from the list of available components displayed on the Component Catalog window of the PSoC Creator IDE. Any necessary wiring between the components is also performed in the schematic.

In the BLE component, it is necessary to create a GATT service and the characteristics of each sensor variable that this peripheral station will send to the central station. The characteristics are defined in the tab "Profiles". The BLE component presents a list of predefined characteristic that can be used. However, if these are not appropriate, others can be created using the "Custom Characteristic" option. For the connection between the central station and the peripheral station to be established automatically, it is necessary to include the universally unique identifier (UUID) of the service in the advertisement packet. This setting is available in the "GAP Settings" tab of the BLE component.

B. Data Collection Protocol

A polling protocol was implemented to regulate the communication between the battery/traction system and the respective BLE module (peripheral station) via UART. The master (BLE module) polls the slave with the same period defined for the Timer component of the BLE module, using a 1-byte frame containing a frame type field, which indicates the content requested from the sensor system. The response frame sent by the battery system is composed by the same frame type, followed by a sample from each of its six sensors. Similarly, the response frame sent by the traction system is composed by the frame type followed by a sample from each of its six sensors. An error message frame is sent if the sensor system is not able to send the requested data. The samples included in the data frame are compact 16-bit integer values generated by the analog-to-digital converters (ADC) of the sensor system. These values are converted into less compact real-valued floating-point numbers when received by the mobile app, for presentation to the user. The protocol also defines a command poll frame, used in the context of the smart charging system, which contains the frame type followed by a data field that indicates the new value of the charging current.

V. Residential WSN Development

The BLE sensor node (peripheral station) of the residential WSN was developed in a similar way to the sensor nodes of the IVWSN. The central station (gateway BLE/Wi-Fi) is based on a Raspberry Pi 3. A program for this platform was developed to read data from the residential BLE sensor node and upload to the Firebase database. It was developed in Java using the IntelliJ IDEA IDE, version 2017, and uses the TinyB library for BLE communication and the Firebase Admin SDK for data upload to Firebase. The development was made using the Raspbian Jessie operating system.

A. Smart Charging Control

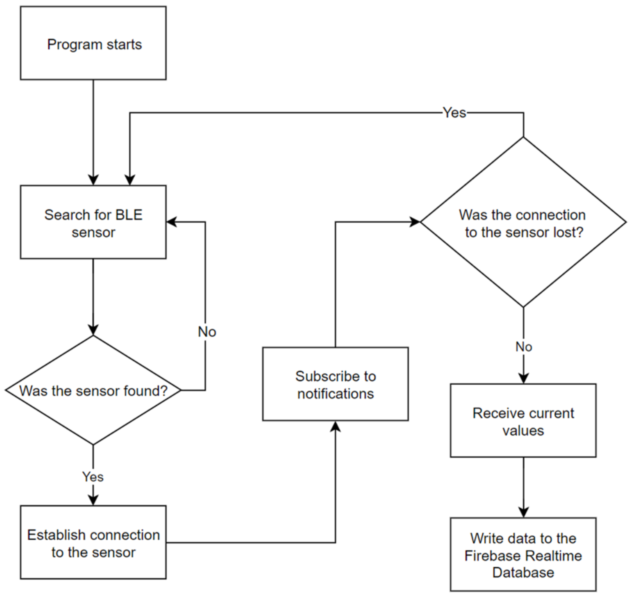
This smart charging system operates under the assumption that the current that is used to charge the vehicle is drained from the same phase, in case of a three-phase installation, of the other appliances of the home. Therefore, in order to avoid the triggering of the circuit breaker of the residence, it is necessary to ensure that the electric vehicle's battery charging current ($I_{charging}$) plus the home current consumed by the appliances (I_{home}), do not exceed than the maximum allowed current (I_{max}), as shown in (1):

$$I_{charging} + I_{home} < I_{max} \tag{1}$$

This restriction means that charging current for the electric vehicle needs to be updated in real time, according to (1), every time there is a change in the RMS value of the home current, which is read by the respective BLE sensor node. In a common scenario, the home current is measured inside of the user's home, but the vehicle's battery charging current is controlled (by the smart charging system) at the garage, which can be located far away. This scenario poses a challenge to the communication system, which needs to provide sufficiently low delay between these points. Therefore, the suitability of the architecture proposed in Fig. 1 to satisfy the delay requirements of the smart charging system is evaluated experimentally in Section VII.b.

B. Home Gateway Application

The Raspberry Pi application uses the BLE notification mechanism to receive the home current data from the BLE sensor node. The life cycle of this application is as follows: 1. Discover the BLE device that provides the current service; 2. Establish connection to the device; 3. Subscribe to the notifications of change of current value; 4. Write the new value to Firebase. Fig. 3 represents the behavior of this application.



Program starts
Yes
Search for BLE sensor
Was the connection to the sensor lost?
No
No
Was the sensor found?
Subscribe to notifications
Receive current values
Yes
Establish connection to the sensor
Write data to the Firebase Realtime Database

Fig. 3. Flowchart of the Raspberry Pi application of the residential WSN.

VI. Mobile App Development

This section describes the implementation of the mobile app, developed using Android, which communicates with the IVWSN sensor nodes using the BLE protocol and plays the role of vehicle's HMI when the device is used inside of the vehicle (gateway mode), and also interacts with the Firebase database either inside or outside of the vehicle (client mode).

A. Configuration of Dependencies

The Android application requires the installation of some dependencies for proper operation. Dependencies are added to the *build.gradle* file generated by Android Studio when the application is created. Three dependencies were installed for Firebase and other three for the Google Play services. Then, the Google Play services plugin was added to the *build.gradle* file of the root project and activated in the last line of this file. It is also necessary to place the *google-services.json* file (which is generated by the Firebase console) in the app module folder. Finally, it was also necessary to enable the Google Maps Application Programming Interface (API). To do this, a key is created in the Google API Console and then the app package name and the SHA-1 digital signature are added to *AndroidManifest.xml*.

B. Application Architecture

The application uses the Model–View–Presenter (MVP) architecture for the presentation layer and a mixed package division between functionalities and layers. In the first level of the packages, the division is made by layer (data layer, graphical user interface (GUI) and services). In the second level, it is made by functionality: battery information, location and profile settings, for example. The function of each package is as follows: data - contains Plain Old Java Objects (POJOs) that represent the application data (battery, traction system, location) and repositories to access the same data; service - contains the Bluetooth service and classes responsible for managing the connections and data collection; ui - contains classes responsible for the GUI (Activities and Fragments); util - contains utility classes that help reduce code repetition in other classes; view - contains custom Views. Within the "ui" package there are other packages that define the application functionality. Each of these packages includes the View and Presenter classes of the MVP pattern and other secondary classes responsible for the application's GUI.

C. Data Communication

When the user enters the app for the first time, he is asked to add at least one vehicle (model and the license number), and then the vehicle is added to the Firebase database.

When the mobile device is configured to operate as the vehicle's gateway, the user may click on the Bluetooth icon of the application toolbar to start communication with the vehicle's sensor nodes. By clicking on this icon, the Bluetooth service (BleService) is started and a notification appears on the screen of the Android device. When the Bluetooth service is called, it starts a Thread to search for devices that implement the services defined in the BLE configuration. The search for these devices is made by the BleScanner class. The user does not need to choose the devices to which to connect. When a device is found, BleScanner passes the information to the BleConnectionManager class to establish a connection. As soon as the list of characteristics of the service is obtained, notifications are requested for each one.

When a notification is received, the BleDataSaver class interprets the data and saves it both in the local cache and in Firebase if the value has changed. Since a 16-bit integer ADC sample representing the measured value is received for each variable, a conversion formula is applied to obtain the real value of the measurement. To avoid excessive data consumption costs, the Firebase upload interval can be configurable with a larger value than the BLE data sending interval (500 ms by default) in the mobile app.

The mobile app also updates the electric vehicle's location (every 10 s), using Google Play services, when operating in the vehicle's gateway mode. The location service uses the LocationUpdater class, which is passed to BleThread. If the user keeps a mobile device inside the vehicle to collect data (gateway mode), he can use another mobile device in client mode to track the vehicle's location and check the data being collected by the BLE sensor nodes through access to Firebase.

D. *Notifications and User Profile*

The notifications implemented in this prototype are related to the battery charge level. When the charge becomes less than 10% or equal to 100%, the Firebase function sends a notification for all tokens registered in the database, which is presented in the notification area of the Android device. Token registration is done in MainPresenter when the user opens the mobile app. To save the token, a key with an empty value is stored. The user profile screen of the mobile app allows the user to enable/disable the notifications and to configure the low battery charge level and the vehicle's data acquisition period.

VII. EXPERIMENTAL RESULTS

A. *Data Collection and Presentation*

This section presents the results of the tests involving the data transfer between the devices that participate in the collection of data from the sensor nodes of the IVWSN, as well as the processing and presentation of the information by the developed mobile app. Fig. 4 shows the information presented by the battery system screen of the mobile app. In this screen, the user can see all the data collected from the electric vehicle's battery system. In the upper part of the screen, the user may also enable the smart charging system, which uses the real time value of the home current provided by the residential WSN to calculate the current available for charging the battery. If this option is disabled, it is possible to manually control the charging process (on/off) with the option "Charge battery".

The obtained results validate the implementation of the associated tasks, including the acquisition of data frames from the battery system by the attached BLE module, the wireless transmission of the sensor values form the peripheral station to the central station, the conversion of the ADC samples to the corresponding real values (in V, A and ºC), and the display of the values, in real time, in the battery system screen of the mobile app. Similar tests were successfully performed for the traction system, validating the corresponding implementation.

B. *Smart Charging Delay*

The total communication delay, from the moment there is a change in the value of the home current measured by the residential WSN until the new value reaches the vehicle's battery system, is an important parameter for the proper operation of the smart charging system, because, if this delay is excessive, the battery charging current may not decrease fast enough, causing the triggering of the circuit breaker.

In order to measure this delay, the test setup presented in Fig. 5 was conceived and implemented. A personal computer (PC) takes the role of both the Raspberry Pi, as the sender of the home current data, and the battery system, as the final receiver of this information. This configuration eliminates errors due to clock differences, since the same machine is used as sender the receiver. The only delay component not covered by this configuration is the BLE medium access delay at the residence, but since this value can be configured to a maximum value as low as 7.5 ms, it is negligible when compared to the total delay.

In this scenario, tests were performed both for the mobile app connected via Wi-Fi and mobile data. The PC generated and sent 1000 current values. The tests were made using a Nexus 5X mobile device near a Wi-Fi router and with 4G connection. The PC had an Ethernet cable connection to the Internet Router, which provided 100 Mbps of upload/download

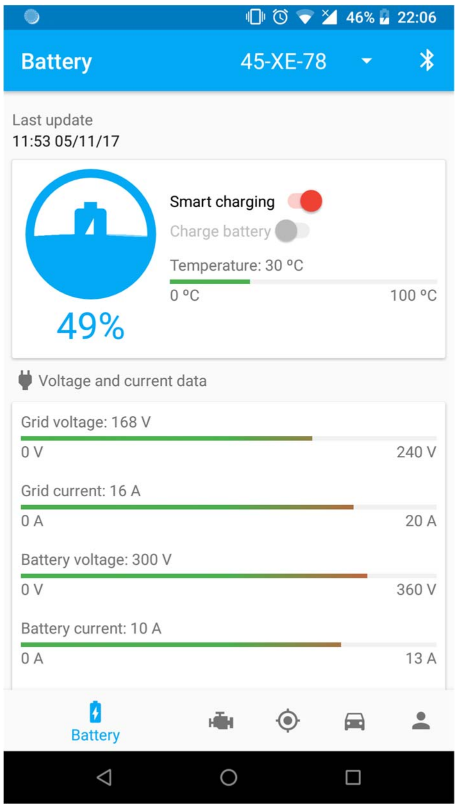

46% 22:06
Battery 45-XE-78
Last update
11:53 05/11/17
Smart charging
Charge battery
Temperature: 30 ºC
0 ºC 100 ºC
49%
Voltage and current data
Grid voltage: 168 V
0 V 240 V
Grid current: 16 A
0 A 20 A
Battery voltage: 300 V
0 V 360 V
Battery current: 10 A
0 A 13 A
Battery

Fig. 4. Battery system information screen.

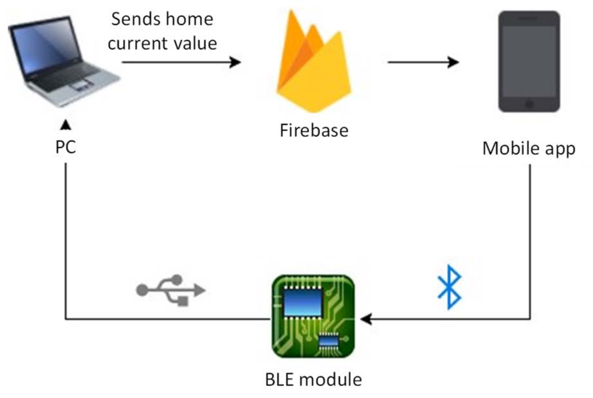


Fig. 5. Test setup for calculation of the smart charging system delay.

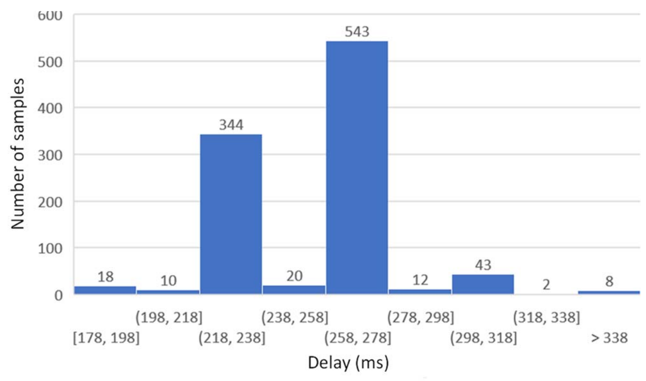


Fig. 6. Distribution of the smart charging delay for the test where the mobile device connected to the Internet through a Wi-Fi router.

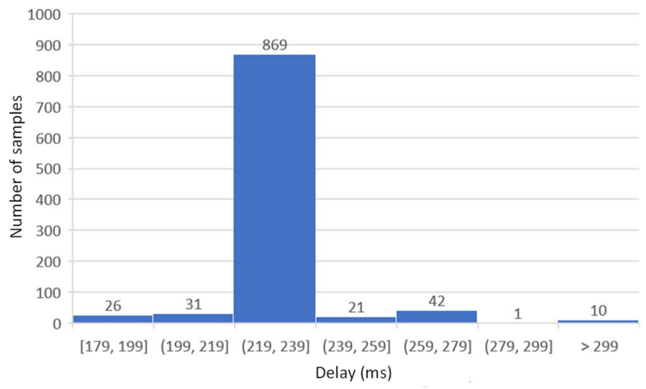


Fig. 7. Distribution of smart charging delay for the test where the mobile device used a 4G connection to the Internet.

access. No other programs were using intensively the connection to the Internet. Table I shows the statistics for the measured delay, in ms, for the Wi-Fi and 4G connections, showing the minimum, mean and maximum values, as well as the standard deviation

TABLE I. MEASURED VALUES FOR THE SMART CHARGING SYSTEM DELAY.

Connection	Min. (ms)	Mean (ms)	Max. (ms)	Std. Dev. (ms)
Wi-Fi	178	257.6	1081	56.31
4G	179	230.4	991	50.97

Fig. 6 and Fig. 7 present histograms representing the distribution of the delay using Wi Fi and 4G, respectively. Table I and these figures show that better results were obtained when the mobile app operated with an Internet connection via 4G in this case, but even for the connection with worse results (Wi-Fi), less than 1% of the messages had a delay higher than 338 ms, and the maximum delay was 1081 ms, which is still enough for the system to quickly adjust the current available for the battery charging without triggering the main circuit breaker of the residence. As a safeguard for the case where the delay exceeds an acceptable value, which may occur when the Internet connection becomes unreliable, the smart charging system may adopt a procedure where the charging current is automatically reduced if a home current update is not received within a predefined amount of time, until a normal connection is reestablished.

VIII. CONCLUSIONS

This paper described the development and test of a smart IoT system that allows the users to monitor and control parameters of their electric vehicles using a mobile app. Inside of an electric vehicle, the app may communicate directly with the sensors nodes through an intra vehicular WSN. Outside of the vehicle, the user may access the data through an online database. The experimental results achieved for this system validated its overall implementation. In the future we intend to increase the functionalities of the IVWSN with the development of new sensors and the integration with other vehicles. We also intend to explore new possibilities for the BLE-based residential WSN, with the deployment of more sensor nodes and evaluation of the performance using the emerging Bluetooth mesh networking specifications.

ACKNOWLEDGMENT

This work is supported by FCT with the reference project UID/EEA/04436/2013, COMPETE 2020 with the code POCI-01-0145-FEDER-006941.

REFERENCES

[1] C. Buratti, A. Conti, D. Dardari, and R. Verdone, “An overview on wireless sensor networks technology and evolution,” Sensors, vol. 9, no. 9, Aug. 2009, pp. 6869-6896.

[2] N.D. Lane et al., “A survey of mobile phone sensing,” IEEE Communications Magazine, vol. 48, no. 9, Sept. 2010, pp. 140-150.

[3] E. Macias, A. Suarez, J. Lloret, “Mobile Sensing Systems,” Sensors, vol. 13, no. 12, 2013, pp. 17292-17321.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of things: A survey on enabling technologies, protocols, and applications,” IEEE Communications Surveys & Tutorials, vol. 17, no. 4, 2015, pp. 2347-2376.

[5] S. Kamath and J. Lindh, “Measuring Bluetooth Low Energy power consumption,” Application Note AN092, Texas Instruments, 2012, pp. 1-24.

[6] J.R. Lin, T. Talty, and O. Tonguz, “On the potential of Bluetooth Low Energy technology for vehicular applications,” IEEE Communications Magazine, vol. 53, no. 1, Jan. 2015, pp. 267-275.

[7] R. Want, B. N. Schilit, and S. Jenson, “Enabling the Internet of Things,” IEEE Computer, vol. 48, no. 1, Jan. 2015, pp. 28-35.

[8] R.B.C. Silva, J. A. Afonso, and J. L. Afonso, “Development and test of an intra-vehicular network based on Bluetooth Low Energy,” World Congress on Engineering 2017, July 2017, pp. 508-512.

[9] M. Siekkinen, M. Hiienkari, J.K. Nurminen, and J. Nieminen, “How low energy is Bluetooth Low Energy? Comparative measurements with ZigBee/802.15.4,” IEEE WCNCW Wireless Communications and Networking Conference Workshops, Apr. 2012, pp. 232-237.

[10] J.R. Lin, T. Talty, and O.K. Tonguz, “An empirical performance study of intra-vehicular wireless sensor networks under WiFi and Bluetooth interference,” IEEE GLOBECOM Global Telecommunications Conference, Dec. 2013, pp. 581-586.

[11] Bluetooth SIG, “Specification of the Bluetooth system. Master table of contents & compliance requirements,” Version 4.2, Dec. 2014.

[12] J.A. Afonso, A.J.F. Maio, and R. Simoes, “Performance evaluation of Bluetooth Low Energy for high data rate body area networks,” Wireless Personal Communications, vol. 90, no. 1, Sept. 2016, pp. 121-141.

[13] D. Pedrosa, V. Monteiro, H. Gonçalves, J.S. Martins, and J.L. Afonso, “A case study on the conversion of an internal combustion engine vehicle into an electric vehicle,” IEEE VPPC Vehicle Power and Propulsion Conference, Oct. 2014, pp. 1-5.

[14] R.A. Sousa, “Development of an Internet of Things System for Communication with Electric Vehicles”, MSc Thesis, University of Minho, Dec. 2017.